CLAIMS

1. A method of enabling multiple different operating systems to run concurrently on the same computer, comprising:

selecting a first operating system to have a relatively high priority;

selecting at least one second operating system to have a relatively lower priority;

providing a common program arranged to switch between said operating systems under predetermined conditions; and

providing modifications to said first and second operating systems to allow them to be controlled by said common program.

2. The method of claim 1, wherein switching between said operating systems includes invoking the common program using exception vectors.

3. The method of claim 2, comprising allocating exception vectors to trap calls, thereby to enable invocation of the common program using a trap call mechanism.

4. The method of claim 2 or 3, wherein the first or second operating system invokes the common program by calling an exception vector.

5. The method of claim 4, wherein calling an exception vector to invoke the common program simulates an exception caused by an external event.

6. The method of any preceding claim, wherein the common program preempts the first or second operating system by intercepting exception or interrupt vectors.

7. The method of claim 6, further comprising using a exception handler table containing an array of pointers to intercept exceptions, and activating an exception handler program to preempt the first or second operating system.

8. The method of any preceding claim, wherein the common program is operable in real mode.

9. The method of claim 8, comprising preempting the first or second operating system by the common program, and switching to real mode when preempting the first or second operating system.

10. The method of claim 8, comprising invoking the common program by the first or second operating system, and switching to real mode when invoking the common program.

11. The method of any preceding claim, comprising enabling hardware interrupts throughout the operation of the second operating system except during the operation of subroutines that save machine state.

12. The method of any preceding claim, in which the first operating system is a real time operating system.

13. The method of any preceding claim, in which the second operating system is a non-real time, general-purpose operating system.

14. The method of any preceding claim, in which the second operating system is Linux, or a version or variant thereof.

15. The method of any preceding claim, in which the common program is arranged to save, and to restore from a saved version, the processor state required to switch between the operating systems.

16. The method of any preceding claim, in which processor exceptions for the second operating system are handled in virtual fashion by the common program.

17. The method of any preceding claim, in which the common program is arranged to intercept some processor exceptions, and to call exception handling routines of the first operating system to service them.

18. The method of claim 17, in which the processor exceptions for the second operating system are notified as virtual exceptions.

19. The method of claim 18, in which the common program is arranged to call an exception handling routine of the second operating system corresponding to a said virtual exception which is pending.

20. The method of any preceding claim, further comprising providing each of said operating systems with separate memory spaces in which each can exclusively operate.

21. The method of any preceding claim, further comprising providing each of said operating systems with first input and/or output devices of said computer to which each has exclusive access.

22. The method of claim 21, in which each operating system accesses said first input and/or output devices using substantially unmodified native routines.

23. The method of any preceding claim, further comprising providing each of said operating systems with access to second input and/or output devices of said computer to which each has shared access.

24. The method of claim 23, in which all operating systems access said second input and/or output devices using the routines of the first operating system.

25. The method of claim 24, in which the common program provides trap call mechanisms, to control the operation of the second operating system, and/or event

mechanisms to notify the first operating system of status changes in the second operating system.

26. The method of any preceding claim, further comprising combining said operating systems and common program into a single code product.

27. The method of any preceding claim, further comprising embedding said operating systems and common program onto persistent memory on a computer product.

28. A development kit computer program product comprising code for performing the steps of any preceding claim.

29. A computer program product comprising code combined according to claim 36.

30. A computer system comprising a CPU, memory devices and input/output devices, having executing thereon computer code comprising;

a first operating system having a relatively high priority;

a second operating system having a relatively lower priority; and

a common program arranged to run said operating systems concurrently by switching between said operating systems under predetermined conditions.

31. A computer system according to claim 30, arranged to run said first and second operating systems concurrently using the method of any of claims 1 to 27.

32. The system, product or method of any preceding claim in which the computer has a Reduced Instruction Set architecture.